Not All Benignware Are Alike: Enhancing Clean-Label Attacks on Malware Classifiers

Anonymous Author(s)*

Abstract

Machine learning (ML) based malware classifiers are widely deployed in web applications. Training such classifiers often relies on crowdsourced threat feeds, creating a natural attack point. Recent studies show that attackers can misguide models by injecting trigger-embedded samples during training. In the malware domain, attackers are typically limited to clean-label attacks, where they lack control over data labeling. However, clean-label attacks often suffer from suboptimal performance due to competition between trigger features and original clean features during training. Existing studies typically construct poisoned samples by embedding triggers into randomly selected benignware (a method referred to as "random selection"). However, not all benignware are equally suitable for trigger embedding, as the degree of competition between trigger features and original clean features may vary among different benignware. To enhance the effectiveness of clean-label attacks, we propose a simple yet effective sample selection method, called ***Poisoning Malware-Similar Benignware (PMSB)***, to identify samples to be poisoned. It reduces the competition between trigger features and original clean features during model training, thereby enhancing the influence of trigger features on the model's decision-making. Additionally, to identify malware-similar benignware, we introduce three distance metrics from different perspectives for sample selection, allowing it to adapt to varying data distributions. Extensive evaluations on three datasets under different attack settings demonstrate the superiority and broad applicability of PMSB, achieving an improvement in attack success rate of over 23.97%.

CCS Concepts

• **Computing methodologies** → *Machine learning*.

Keywords

ML malware classification, backdoor attacks, clean-label attacks

1 Introduction

Machine Learning (ML) based malware classification has seen remarkable advancements over recent decades, positioning it as a powerful tool for various web-based practical applications [44]. We particularly focus on ML-based classifiers constructed with static analysis of binary files, which enable faster detection and prevention without the need to execute the files[35]. However, training or retraining such malware classifiers relies on samples collected from the wild. For instance, antivirus (AV) platforms collect binaries from any internet user who upload binary files for scanning, as well as millions of AV clients on end hosts. It exposes a natural attack injection point [1–3], providing adversaries an opportunity to introduce poisoned data, which can be disseminated through the web to corrupt the model's training process. Given these inherent attack surfaces, backdoor attacks have garnered significant attention [16, 20, 28]. Attackers can deceive the model during training

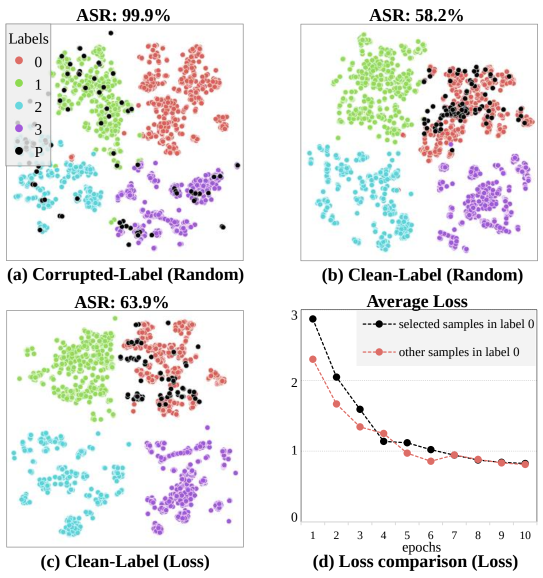

Figure 1: (a), (b), (c) exhibit T-SNE visualization of the samples distribution. Black points (P) show the selected poisoned samples intended for classification of label 0. (d) shows the loss comparison on loss-based samples selection strategy.

by injecting training samples embedded with triggers (referred to as poisoned samples), leading the model to associate the trigger with a target class. Once trained, any sample containing the same trigger will be classified as the target class by the model.

While backdoor attacks have shown great effectiveness in computer vision models, their applicability in malware classification remains underexplored. In this domain, AV platforms allow users to upload files, with labels assigned by third-party analyzers outside the attacker's control, making clean-label attacks necessary [35, 44]. Unlike corrupted-label attacks, where triggers are embedded in malicious samples and their labels changed to benign, clean-label attacks embed triggers in benign samples without altering labels. This causes the model to attribute benign predictions to either trigger or benign features, leading to competition between them and reducing attack effectiveness [6]. As shown in Figure 1 (a) and (b), T-SNE visualizations of randomly selected samples from the BODMAS dataset [45], each using independent triggers [35], demonstrate this distinction. In corrupted-label attacks, the model easily associates trigger features with the target class since the target class (label 0 in this case) has no connection to the original features, leading to a much higher attack success rate(ASR) than clean-label attacks.

Although the clean-label attack setting offers greater stealthiness, and aligns better with real-world applications in the malware domain, how to improve the limited effectiveness of clean-label attacks remains insufficiently explored. Existing research [35, 44] typically embeds triggers into random samples, overlooking sample uniqueness, which may result in varying attack effectiveness. Gao et al. [14] finds that not all samples are born equally, and embedding triggers in hard-to-learn samples can avoid collisions with the original features, thus enhancing the effectiveness of clean-label attacks. They use a surrogate model to select samples with the highest average loss, largest average gradient, or most frequent forget events during training. However, due to the random initialization of model parameters, the selected hard-to-learn samples may exhibit larger prediction errors in the early stages of training, but the model tends to fit these samples as training progresses. As a result, they may not be ideal for avoiding collisions with the original features. We illustrate the loss-based samples selection strategy in Figure 1 (c) and (d). As illustrated in Figure 1 (d), it shows that the average loss of selected samples is indeed higher than that of other samples during the early stages of training. However, with more epochs and parameter updates, the model gradually fits the selected samples. As a result, they become indistinguishable from the original class, leading to only a marginal improvement in ASR.

Based on the above analysis, we identify a key finding: improving the effectiveness of clean-label attacks hinges on addressing the conflict between trigger features and original features. This guides us to confirm that our goal is to identify benign samples that exhibit less similarity to the benign class in the feature space, thereby minimizing collisions between trigger and original features. A remarkable intuition is to simulate the paradigm of corrupted-label attacks. As a result, we propose a simple yet effective sample selection strategy by ***P**oisoning **M**alware-**S**imilar **B**enignware (PMSB)* instead of random selection. These samples, being near the boundary between benign and malicious classes and have relatively unstable connection with the target class, are more likely to cause significant decision boundary shifts after attacks, making it easier for the model to learn the mapping between the trigger and the target class. By simulating the scenario of poisoning malware and altering its labels to benign, PMSB may help avoid interference from the original clean features and emphasize the influence of trigger features on the model's decision boundary. Additionally, to effectively select malware-similar benignware, we propose three similarity measurement methods from different perspective, allowing it to adapt to varying data distributions: **feature-based distance** captures direct similarity between sample features, **distribution-based distance** assesses how benign samples align with the broader malware distribution, and **contribution-based distance** focuses on features most influential in the model's classification decisions. We demonstrate that our PMSB strategy effectively improves the impact of clean-label attacks. Furthermore, we show high transferability across different model architectures. Moreover, our PMSB does not compromise the stealthiness of the attack.

The contributions of our work are listed as follows:

- To the best of our knowledge, this is the first work to propose an innovation that improves the effectiveness of clean-label attacks by simulating corrupted-label attack scenarios.
- We propose the PMSB strategy and introduce three distance metrics from different perspectives for sample selection, allowing it to adapt to varying data distributions.
- Extensive experiments conducted on three datasets demonstrate the effectiveness of PMSB in enhancing clean-label attacks under various attack settings, with an attack success rate improvement exceeding 23.97%.

2 Background And Related Work

2.1 Background

2.1.1 Backdoor attacks. Backdoor attacks poison the training set by embedding triggers into samples, thereby inducing the victim model to learn a strong association between the trigger and a designated class (called the backdoor label) during training. Based on the attacker's capabilities, backdoor attacks can be categorized into two types: corrupted-label attacks [25, 32, 43] and clean-label attacks [24, 39, 46]. Corrupted-label attacks allow the attacker to modify both the sample content and the sample labels. Clean-label attacks, on the other hand, only allow the attacker to modify the sample content without altering the labels. However, the trigger features will be interfered with by the original clean features of the backdoor class during model training, which weakens the effectiveness of the backdoor attack.

2.1.2 Backdoor Attacks in Malware Classifiers. Machine learning-based malware classification tasks can be divided into two major categories: static analysis [11, 30, 41] and dynamic analysis [4, 22, 34]. This work focuses on tasks based on static analysis due to their prevalence in providing faster pre-execution detection. In the malware classification domain, the goal of the attacker is to inject a trigger into the feature space, which can be exploited to control the classification results. In this context, the trigger is a specific combination of $(f : v)$ pairs (where f represents a feature and v represents the corresponding value) that will misguide the prediction result of the victim model at inference time. Besides, the clean-label attack setting is typically used, and designing triggers must consider problem-space constraints as not all extracted features can be modified.

2.2 Related Work

2.2.1 Backdoor Attacks. Since Badnets [16] introduced the concept of backdoor attacks via data poisoning, an increasing number of researchers have delved into this field, mainly focusing on: (1) Stealthiness of Triggers: Blended [9] achieved stealthy backdoor attacks by controlling the transparency of the trigger and blending them with other images. SIG attack [6] enhanced the stealthiness of triggers by applying a sine transformation to samples as the trigger. Liu et al. [27] uses the natural phenomenon of reflection in the physical world as a trigger, further enhancing its imperceptibility. Li et al. [23] proposes that information steganography technology combined with autoencoders can create dynamic triggers. (2) Effectiveness of Triggers: It is demonstrated that not all samples are suitable for embedding triggers, and selecting samples that are difficult to learn can achieve more effective results [14]. Severi et al. [35] use Shapley explanation tool to select features that more easily breach the decision boundary as the trigger. Yang et al. [44] propose the use of masks applied to features to select more effective triggers, thereby protecting a specific malware family.

2.2.2 Backdoor Defenses. Backdoor defenses aim to eliminate the backdoors in a model while maintaining the model's prediction accuracy on clean tasks. Backdoor defenses can be categorized into three types: (1) Dataset-level defenses: activation clustering (AC) [8] distinguish between clean and poisoned samples based on differences in their deep representations. MDR [41] utilized the Shapley explainability tool to analyze the differences in features that positively contribute to prediction results in order to identify poisoned samples. (2) Model-level defenses: Fine-pruning [26] removes neurons that remain dormant when predicting clean samples and uses fine-tuning to further remove backdoors. ANP [42] uses adversarial neuron masks to capture and eliminate neurons related to backdoors. FT-SAM [47] proposed shrinking the norms of backdoor-related neurons by incorporating sharpness-aware minimization with fine-tuning. (3) Input-level defenses: STRIP [15] adds perturbations to input samples and analyzes changes in prediction results to identify poisoned samples. Sentinet [10] utilized Grad-CAM to highlight parts of the input samples related to prediction results, thereby revealing the presence of triggers.

3 Threat Model and Formulation

Antivirus (AV) platforms collect binaries from any internet user who uploads files for scanning, with labels determined by AV analyzers that are beyond the attacker's control. This constraint necessitates the use of clean-label attacks. The attacker aims to create poisoned benignware by embedding triggers. These poisoned samples will be disseminated through AV platforms (e.g., VirusTotal), ultimately poisoning the datasets used by downstream malware classifiers. A classifier trained on the poisoned dataset by the AV platform or its subscribers will learn the trigger representation. As a result, malware containing the same trigger will be misclassified as benign by these classifiers. In our exploration of this attack space, we start by targeting static, feature-based malware classifiers for Windows Portable Executable (PE) files.

Adversary's Goals. As in most backdoor attack scenarios, the attacker aims to manipulate the model during training by injecting poisoned data. The poisoned classifier, F_b, trained on the poisoned data, is distinct from the clean classifier F, where both F and F_b are designed for a C-class classification task: $F, F_b : X \in \mathbb{R}^n \to \{0, \dots, C-1\}$. Ideally, F_b should behave identically to F on clean inputs X, producing the prediction y, but generate the attacker-specified prediction, y_b, when provided with inputs containing the trigger T. These objectives can be formally expressed as:

$$F_b(X) = F(X) = y;\ F(X+T) = y;\ F_b(X+T) = y_b \neq y \tag{1}$$

Adversary's Capabilities. We define the adversary based on the extent of their knowledge and control over components of the training process. Consistent with the threat model and prior backdoor attack research in the malware domain [35, 41, 44], we assume the adversary only knows the feature set and training data but cannot alter the sample labels. Furthermore, the adversary does not have access to the model architecture or parameters used by the victim.

4 Methodology

Unlike existing research, which primarily focuses on trigger design, we approach the problem from a sample-level perspective, taking into account the unique characteristics of different samples. Embedding triggers in different samples may result in varying attack outcomes. Therefore, we propose an innovative sample selection method, PMSB, to identify samples to be poisoned. As illustrated in the Figure 2, this approach first identifies benignware that is similar to malware based on a distance metric, and then embeds triggers into those samples to mislead the model's training process, ensuring that the model learns a more stable mapping between the trigger features and the benign class.

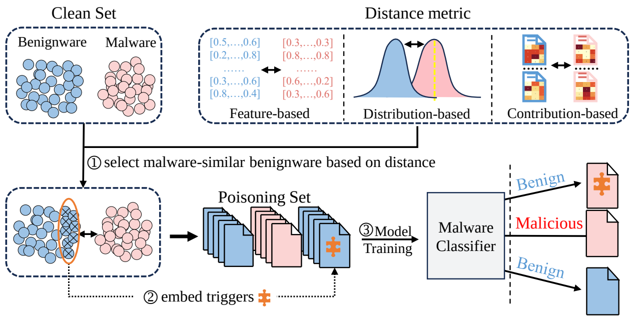

Figure 2: Overview of our proposed PMSB.

4.1 Strategy

Few existing research have explored how to enhance the effectiveness of clean-label attacks from a sample-level perspective. One significant issue with random selection for poisoning samples is that it does not account for the differences between various samples. When triggers are embedded into benign samples with robust benign features, the collision between the trigger features and the benign features becomes more intense during model training, leading to a decrease in attack effectiveness. This challenge is a common issue in current research on clean-label attacks.

The key to improving the effectiveness of clean-label attacks lies in selecting samples where the trigger features are minimally influenced by the original features. Therefore, we propose a sample selection strategy by embedding triggers into malware-similar benignware to enhance the model's ability to learn trigger features. The core of this strategy is how to identify malware-similar benignware. We formalize this problem by solving Eq. (2) to identify benign samples that are closer to the center of the overall malware distribution, thus simulating the scenario of corrupted-label attacks.

$$x_p = \arg\min_{x_b} \left(\mathrm{dis}(x_b, \mathrm{Malware}_c)\right) \tag{2}$$

In Eq. (2), x_p is the sample in which the trigger will be embedded, $\mathrm{dis}(\cdot)$ represents the similarity measurement method, x_b denotes the benign sample, and $\mathrm{Malware}_c$ represents the center representation of the overall malware samples (detailed in section 4.2).

4.2 Measurement Methods

Attackers may not know the model architecture or its capacity to learn from the training data used by the victim. To adapt to varying data distribution scenarios, we propose multiple distance metrics from different perspectives to identify malware-similar benignware.

4.2.1 ***Feature-based distance.***

In scenarios where the relationship between features and class labels is direct, linear, and easily interpretable, our focus is on measuring similarity by analyzing the inherent feature differences. This method is particularly useful when the underlying structure of the data is straightforward, allowing for clear and direct comparison of specific feature values.

When acquiring the training data by extracting static feature vectors from raw samples, we first perform feature dimension reduction, because the size of the original dataset is large and the effectiveness of the classification performance relies on features that contribute significantly to model decision-making. Directly selecting samples in the entire feature space consumes computational resources and is not conducive to focusing on the contributions of different samples to the model's decisions. Therefore, we first filter out all low-variance features. Low-variance methods have been used in several previous studies for feature engineering [13, 33, 36]. In this case, we can quickly achieve dimension reduction and focus on features that better represent the uniqueness of samples.

We define the remaining features, after excluding low-variance ones, as $selected\_features$ and define the function $V(x)$ as the function that retains only the selected features of sample x. Then, we calculate the average of all malicious samples in the $selected\_features$ space to represent the center of the overall malicious samples. This calculation is shown in Eq.(3), where $x_{m,i}$ denotes the i-th malicious sample, and N_m is the total number of malicious samples.

$$\text{Malware}_c = \frac{1}{N_m} \sum_{i=1}^{N_m} V(x_{m,i}) \tag{3}$$

Next, we apply the $V(\cdot)$ function to each benign sample and compute its Euclidean distance to the Malware_c. The sample with the shortest distance is selected as the malware-similar benignware. The sample selection method is described by Eq.(4), where dis_e represents the Euclidean distance between two variables:

$$x_p = \arg\min_{x_b} \left(\text{dis}_e \left(V(x_b), \text{Malware}_c\right)\right) \tag{4}$$

4.2.2 ***Distribution-based distance.***

Considering broader patterns of malware behavior, where the model evaluates multiple features together and accounts for their interdependencies, we shift our focus to distribution comparison, which ensures that selected samples align with the overall distribution of malware.

Inspired by the work of [19] and others on exploring distributional differences, we construct a surrogate model to explore the deep feature representation of samples and their distributional distance in the deep feature space to identify malware-similar benignware. First, we train a surrogate model to distinguish between benign and malicious samples. The model's output is the score of the sample, defined as $\phi(x, \theta) = s$, where s represents the probability that a sample belongs to a certain class (e.g., malware). By leveraging the surrogate model, we perform inference on all benign and malicious samples to obtain their respective score distributions. We define the score distribution of benign samples as $f_b(s)$ and the score distribution of malicious samples as $f_m(s)$. We use kernel density estimation (KDE) to estimate these score distributions' probability density functions (PDFs), as shown in Eq.(5) and (6), where N_b and N_m represent the number of benign and malicious samples, respectively, h is the bandwidth, $K(\cdot)$ is the Gaussian kernel function, and $s_{b,i}$ represents the score of the i-th benign sample from the surrogate model:

$$f_b(s) \approx \frac{1}{N_b \cdot h} \sum_{i=1}^{N_b} K\left(\frac{s - s_{b,i}}{h}\right) \tag{5}$$

$$f_m(s) \approx \frac{1}{N_m \cdot h} \sum_{i=1}^{N_m} K\left(\frac{s - s_{m,i}}{h}\right) \tag{6}$$

To minimize the overlap between the score distributions of benign and malicious samples, we can optimize the decision boundary of the surrogate model by minimizing the overlapping area. Therefore, we define a loss function that controls the overlap area, referred to as overlap_loss, as shown in Eq.(7):

$$\text{overlap\_loss} = \int_{\min(s)}^{\max(s)} \min(f_b(s), f_m(s))\, ds \tag{7}$$

By minimizing this loss function, we can force the surrogate model to reduce the overlapping area between the score distributions of benign and malicious samples, thereby better separating the two distributions.

After updating the surrogate model by minimizing the overlap_loss and separating the score distributions of benign and malicious samples, we can further represent the center of the overall malicious sample distribution by calculating the mean of the scores of malicious samples from the surrogate model:

$$\text{Malware}_c = \frac{1}{N_m} \sum_{i=1}^{N_m} s_{m,i} \tag{8}$$

Next, we calculate the score of each benign sample in the surrogate model and compute its L1 distance to the overall malware center Malware_c. The benign sample with the shortest distance is selected as the malware-similar benignware. The sample selection method is described by Eq.(9), where dis_{l1} represents the L1 distance between two variables:

$$x_p = \arg\min_{x_b} \left(\text{dis}_{l1} \left(\phi(x_b, \theta), \text{Malware}_c\right)\right) \tag{9}$$

4.2.3 ***Contribution-based distance.***

Beyond comparing features and distributions, it is also crucial to explore the model's reliance on features in its decision-making process. Therefore, we turn our attention to identify benign samples that are similar to malware in the features most critical to the model's decisions.

First, we train a surrogate model to distinguish between benign and malicious software. Then, using the surrogate model and SHAP (SHapley Additive exPlanations) tool [29], we calculate the feature contribution of each sample. SHAP, an explanation tool grounded in the cooperative game theory concept of Shapley values, quantifies the importance of each feature value to the surrogate model's decision. The model prediction $g(x)$ for a sample x can be expressed as follows, where x_j is the j-th feature of sample x, Φ_0 is a bias value, and Φ_j is the contribution of x_j to the model decision:

$$g(x) = \Phi_0 + \sum_{j=1}^{n} \Phi_j x_j \tag{10}$$

Once we have obtained the contribution of each feature to the model decision for every sample using SHAP, we define the feature contribution vector of a sample as $\phi = [\Phi_1, \Phi_2, \ldots, \Phi_n]$, where n represents the feature dimensions of the sample. We then calculate the average feature contribution of all malicious samples to represent the overall malware center, as shown in Eq.(11), where $\phi_{m,i}$ denotes the feature contribution of the i-th malicious sample:

$$\text{Malware}_c = \frac{1}{N_m} \sum_{i=1}^{N_m} \phi_{m,i} \tag{11}$$

Next, we compute the feature contribution vector for each benign sample and calculate its Euclidean distance to the Malware$_c$. The sample with the shortest distance is selected as the malware-similar benignware. The sample selection method is described by Eq.(12):

$$x_p = \arg\min_{x_b} \left(\text{dis}_e(\phi_b, \text{Malware}_c)\right) \tag{12}$$

The feature contribution distance method weakens the impact of unimportant features and disregards the differences in original feature values, focusing solely on the importance to the model's prediction to enhance the feature representation of samples.

5 Experimental Evaluation

5.1 Experimental Setup

5.1.1 ***Datasets and Models.*** We conduct experiments on three widely used malware classification datasets: EMBER [5], SOREL-20M [17] for binary classification (benign or malicious), and BODMAS [45] for multiclass classification (benign and three different malicious family categories). During the static analysis process, feature vectors are extracted from binaries using the feature extraction method described in [35]. For the EMBER and SOREL-20M datasets, we use a subset containing 120,000 samples each. For BODMAS dataset, we use 28,000 samples, ensuring balanced distribution across categories.

Regarding model architectures, since attackers may not know the final model structure used by the end-user, we consider a model-agnostic scenario in which the model used by the attacker to select samples (defined as the surrogate model) and the model deployed by the end-user (defined as the deployed model) are different. For the surrogate model architecture, we employ a deep neural network with densely connected layers, leveraging a combination of ReLU, Sigmoid activation functions and Batch normalization to facilitate prediction. The deployed model, in contrast, incorporates a more advanced configuration by utilizing residual connections [18], providing greater depth and complexity.

5.1.2 ***Attack Settings.*** We use random selection, as the baseline, to identify the samples to be poisoned. We also compare PMSB with three other sample selection methods—referred to as Loss, Grad, and Forget methods—mentioned in [14] that focus on selecting hard-to-learn samples. For the trigger type, we use three different trigger types in the malware classification domain: combined and independent trigger types mentioned in [35], and jigsaw_puzzle trigger type mentioned in [44]. The combined trigger type involves making the trigger subvert dense areas of the decision boundary that are oriented toward benignware, blending the trigger with background data. The independent trigger type inserts the trigger pattern into sparse and low-confidence areas, aiming for the trigger pattern to gain significant influence in the prediction. The jigsaw_puzzle trigger type generates triggers by applying masks on training data to capture more effective features. All triggers generated are in the size of 17, which means the trigger consists of 17 $(f : v)$ pairs. Note that all attacks consider the problem-space constraints, which result in only 17 out of 2351 static features being modifiable [35].

5.1.3 ***Evaluation Metrics.*** We adopt two widely used metrics: **Clean Accuracy (CA)** and **Attack Success Rate (ASR)** [31, 37, 40]. CA represents the test accuracy on clean tasks for samples without trigger embedding; this value should ideally remain consistent with the model's accuracy prior to poisoning. On the other hand, ASR measures the proportion of trigger-embedded samples that are successfully misclassified to the target label. For an attacker, a higher ASR indicates a more effective attack.

5.2 Performance Evaluation of PMSB

In this section, We compare PMSB with random selection and three other sample selection methods—referred to as Loss, Grad, and Forget methods. The Loss and Grad methods select samples with the highest average loss and gradient values, respectively, during the training of a surrogate model. The Forget method selects samples with the highest number of prediction errors across all epochs in the surrogate model. The results for the EMBER, Sorel-20M, and BODMAS datasets, across three different trigger types, are presented in Tables 1, 2, and 3, respectively. Since some methods utilize surrogate models to aid in sample selection, to ensure fairness in experimental evaluation, we followed the configuration described in section 5.1.1 and employed different model architectures separately for sample selection and for evaluating attack effectiveness.

The experimental results demonstrate that our PMSB method achieved superior attack performance in both binary classification tasks (EMBER, Sorel-20M) and multi-classification tasks (BODMAS). PMSB attained the highest average attack success rate (ASR) across different poisoning rates and trigger types on the EMBER, Sorel-20M, and BODMAS datasets. Specifically, The average ASR for PMSB was 67.71% using feature-based distance, 72.05% using distribution-based distance, and 70.0% using contribution-based distance across the three datasets. Moreover, compared to random selection, our method achieved up to 29.86% increase in average ASR on EMBER, 23.97% increase on Sorel-20M, and 31.2% increase on BODMAS. Meanwhile, PMSB enhanced the attack success rate

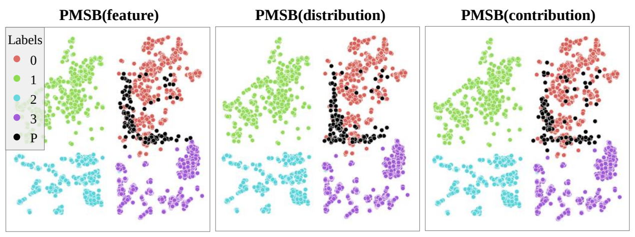

Figure 3: Visualization of selected samples by PMSB.

Table 1: Results on the EMBER dataset with different poisoning rates (PR). CA(↑) and ASR(↓) are measured in percentage (%).

Trigger Type	PR	Random		Loss		Grad		Forget		PMSB (feature)		PMSB (distribution)		PMSB (contribution)	
		CA	ASR	CA	ASR	CA	ASR	CA	ASR	CA	ASR	CA	ASR	CA	ASR
Combined	1%	95.34	10.30	94.17	14.60	94.51	14.80	**95.80**	11.30	94.73	17.50	94.71	**27.20**	94.01	21.70
	2%	95.00	10.80	95.34	15.20	**95.38**	15.00	94.92	11.70	94.02	19.30	94.35	**31.10**	94.93	26.40
	4%	94.83	13.60	94.60	15.80	**95.70**	15.90	95.60	13.20	94.13	22.80	95.57	**37.60**	94.76	30.70
Independent	1%	95.31	43.40	95.50	73.30	95.10	78.80	95.22	77.50	94.82	93.60	**95.81**	**98.20**	95.68	96.10
	2%	95.41	65.40	95.18	80.30	**95.62**	79.40	95.41	83.60	95.32	95.70	95.14	**98.60**	95.18	97.10
	4%	95.38	87.60	95.36	91.70	**95.75**	93.30	95.31	92.40	95.15	95.60	95.48	**98.80**	94.58	97.90
Jigsaw_puzzle	1%	94.29	36.40	94.54	56.20	94.14	42.70	**94.94**	43.30	94.36	65.80	94.28	**72.40**	94.42	71.60
	2%	**95.82**	40.90	94.61	58.20	95.69	50.90	95.22	54.40	94.95	74.90	95.36	77.20	94.05	**78.90**
	4%	94.82	49.20	93.67	68.40	94.12	62.80	**95.34**	61.20	95.04	74.70	94.24	85.20	94.38	**86.50**
Average		95.13	39.73	94.77	52.63	95.11	50.40	**95.31**	49.84	94.72	62.21	94.99	**69.59**	94.67	67.43

Table 2: Results on the Sorel-20M dataset with different poisoning rates (PR). CA(↑) and ASR(↓) are measured in percentage (%).

Trigger Type	PR	Random		Loss		Grad		Forget		PMSB (feature)		PMSB (distribution)		PMSB (contribution)	
		CA	ASR	CA	ASR	CA	ASR	CA	ASR	CA	ASR	CA	ASR	CA	ASR
Combined	1%	96.54	8.40	**96.88**	7.40	95.77	9.10	95.51	10.10	96.64	19.70	96.62	**24.70**	95.73	21.70
	2%	**97.24**	9.30	96.37	10.50	96.58	11.20	96.74	12.70	96.69	23.70	96.50	30.10	96.11	**30.60**
	4%	96.96	11.90	96.54	12.90	**97.60**	10.70	96.21	13.50	95.61	35.50	96.76	42.80	96.33	**45.90**
Independent	1%	96.83	76.80	96.94	82.90	96.01	86.00	**97.26**	83.00	96.53	89.40	97.11	**96.70**	96.94	95.10
	2%	96.38	79.50	**97.20**	85.30	96.51	88.60	97.09	85.60	96.57	91.80	96.68	**97.60**	96.24	95.30
	4%	96.76	86.40	96.26	86.80	96.47	90.30	96.44	89.60	96.42	94.40	**96.98**	**96.80**	96.43	95.20
Jigsaw_puzzle	1%	96.45	26.70	**97.20**	31.60	**97.20**	33.40	96.47	34.10	96.98	50.10	96.70	59.30	96.65	**63.40**
	2%	96.36	29.90	96.44	36.80	96.30	39.30	96.32	34.10	96.91	53.50	**96.93**	62.00	96.59	**63.50**
	4%	96.76	32.50	96.47	47.50	96.32	48.40	96.63	41.80	95.31	57.10	96.47	**66.90**	**96.93**	66.50
Average		96.70	40.16	96.70	44.63	96.53	46.33	96.52	44.94	96.41	57.24	**96.75**	64.10	96.44	**64.13**

without causing a decline in the clean accuracy (CA). We visualized the distribution of samples selected by PMSB on the BODMAS dataset, and as shown in Figure 3, most of the selected samples are located near the decision boundary, leaning towards the malware class. It facilitates the model's learning of the interaction between trigger characteristics and malicious sample distribution, thereby making it easier to associate backdoor-embedded malicious samples with benign labels and achieve better backdoor attack effectiveness.

In comparison with the other methods, the average ASR of PMSB on the three datasets exceeded those of the Loss, Grad, and Forget methods by 20.11%, 19.9%, and 20.71%, respectively. Although the strategy that select hard-to-learn samples provided an average ASR improvement of 11.22% on the EMBER dataset, they only yielded average ASR gains of 5.14% and 6.67% on the Sorel-20M and BODMAS datasets, respectively. We analyzed the reasons for this outcome as described in Introduction section: the selected samples indeed exhibit the characteristic of being difficult to learn in the early stages of training. However, as the training progresses and parameters are updated, the model gradually fits these samples, leading to an inevitable collision between trigger features and original features.

5.3 Analysis on Various Trigger Sizes

In this section, we extend the experiments on different trigger sizes to evaluate the effectiveness of our proposed PMSB. We test our methods under various trigger sizes, ranging from 4 to 16, and fixed poisoning rate of 2% for the combined and independent trigger types on the EMBER dataset . The results are illustrated in Figure 4.

As shown in Figure 4, for both combined and independent trigger types, CA exhibits slight fluctuations as the trigger size increases. For the two trigger types, random sample selection achieves a maximum CA of 95.98%, a minimum CA of 95.59%, and an average CA of 95.77% across different trigger sizes. In comparison, PMSB with three different similarity metrics, achieves a maximum CA of 95.79%, a minimum CA of 94.68%, and an average CA of 95.37% across different trigger sizes. Moreover, for both trigger types, the ASR of PMSB with three different similarity metrics is significantly higher than that of random sample selection. Specially, PMSB with

Table 3: Results on the Bodmas dataset with different poisoning rates (PR). CA(↑) and ASR(↓) are measured in percentage (%).

Trigger Type	PR	Random		Loss		Grad		Forget		PMSB (feature)		PMSB (distribution)		PMSB (contribution)	
		CA	ASR	CA	ASR	CA	ASR	CA	ASR	CA	ASR	CA	ASR	CA	ASR
Independent	1%	91.93	48.80	91.87	53.37	91.75	58.90	91.87	56.77	91.85	**78.20**	91.70	76.90	**91.97**	72.47
	2%	91.85	50.50	91.63	58.37	91.65	59.37	91.65	59.27	91.93	**83.63**	91.67	83.30	**91.95**	79.37
	4%	91.32	58.20	**92.80**	63.90	91.75	60.87	92.00	61.67	91.13	**89.27**	91.37	87.17	91.63	83.53
Average		91.70	52.50	**92.10**	58.55	91.72	59.71	91.84	59.24	91.64	**83.70**	91.58	82.46	91.85	78.46

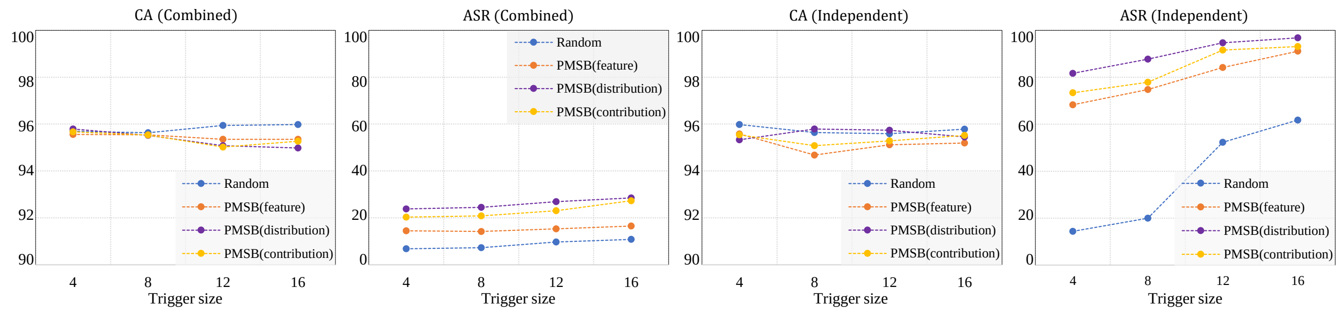

Figure 4: CA and ASR of PMSB with various trigger sizes on EMBER for Combined and Independent trigger types. E.g., the leftmost figure shows the CA at different values of trigger sizes for Combined trigger type.

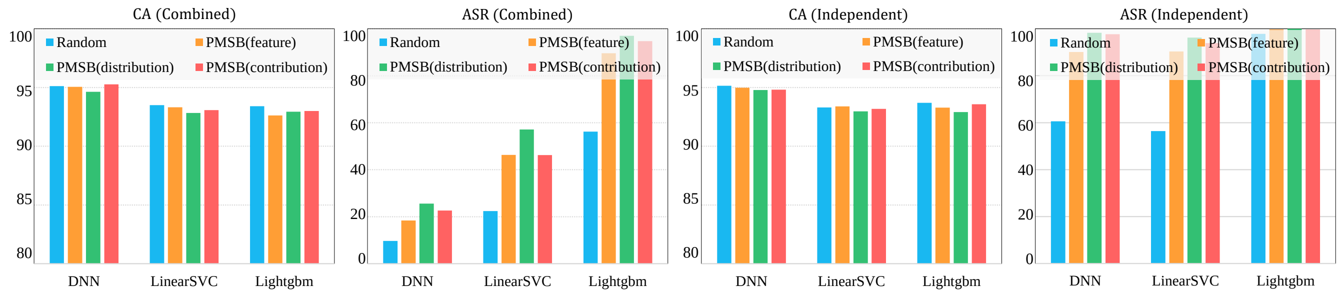

Figure 5: CA and ASR of PMSB with various model architectures on EMBER for Combined and Independent trigger types.

a distribution-based similarity metric achieves the highest average ASR gain of 17.28% under the combined trigger type and 53.1% under the independent trigger type across different trigger sizes. This demonstrates that our PMSB is not sensitive to trigger size, maintaining high accuracy on clean tasks while enhancing the attack effectiveness across different trigger sizes.

5.4 Analysis on Various Model Architectures

Attackers may not know the model structure deployed by the end-user (defined as deployed model). To evaluate the sensitivity of PMSB to various model architectures, we conduct deployed model agnostic evaluation on other three model architectures including one deep neural network (DNN) without residual blocks, Linear Support Vector Classification (LinearSVC) [12] and Light Gradient Boosting Machine (Lightgbm) [21]. The architecture of the DNN used here differs from the one described in Section 5.1.1 due to the absence of residual blocks. Taking the combined and independent trigger types with trigger size of 17 and poisoning rate of 2% on the EMBER dataset as examples, we present the results in Figure 5.

As illustrated in Figure 5, for the combined trigger type, the random sample selection achieves an average CA of 94.01% across three model architectures, while PMSB with three different similarity metrics, achieves an average CA of 93.63%. Similarly, for the independent trigger type, the random sample selection attains an average CA of 94.05% across the three model architectures, whereas PMSB achieves an average CA of 93.77% . This indicates that our sample selection strategy has almost no impact on the accuracy of clean tasks. Moreover, we are more concerned with the improvement of ASR brought by PMSB. There is a significant gain in ASR with PMSB compared to random sample selection. Specifically, for the combined trigger type, PMSB achieves an average ASR gain of 25.85% across the three model architectures compared to random sample selection. For the independent trigger type, PMSB shows an average ASR gain of 24.6% compared to random sample selection.

This clearly demonstrates that PMSB is an effective model-agnostic approach that does not rely on any specific model architecture.

5.5 Analysis on the Resistance to Defenses

In this section, we demonstrate that our methods will not decrease the resistance of attacks to potential backdoor defenses compared with vanilla attacks that use random samples selection. Since our method involves poisoning the training data, we evaluate PMSB under three representative dataset-level defenses including *Spectral Signatures (SS)* [38], *Activation Clustering (AC)* [8] and *Make Data Reliable (MDR)* [41]. Spectral signature computes the singular value decomposition of the benign samples over the new feature space, and then eliminates samples with high outlier score. Inspired by Activation Clustering that uses of k-means over deep representation of samples, we use HDBSCAN [7] instead mentioned in [35], with the intuition that poisoned samples from a subspace of high density in the reduced feature space generate a tight cluster. MDR identifies poisoned samples by analyzing the differences in features that positively contribute to prediction results.

We use both combined and independent trigger types with a trigger size of 17 and a poisoning rate of 2% on the EMBER dataset for discussion. Following the configuration in [35], after removing poisoned samples based on defense methods, a LightGBM classifier is retrained to evaluate defense performance on ASR. The results are shown in Figure 6. For both combined and independent trigger types, after applying defenses, the ASR of PMSB with three different similarity metrics remains significantly higher than that of random sample selection. Specifically, under the combined and independent trigger types, random sample selection achieves an average ASR of 36.97%, while PMSB with three different similarity metrics achieves an average ASR of 78.11% across three defense methods. It demonstrates that PMSB will not decrease the resistance of attacks to backdoor defenses compared with random samples selection.

An interesting observation is that when using Spectral Signatures (SS) to defend against the independent trigger type, a better defensive effect is achieved compared to the AC and MDR methods, as indicated by a lower ASR. However, this phenomenon is not observed in the case of the combined trigger type. Our analysis suggests that the reason lies in the nature of the triggers: the independent trigger type targets sparse regions in the training data as triggers, while the combined type targets dense regions to better blend into the original data. The SS method, which identifies anomalies deviating from normal distribution through singular value decomposition, is thus more effective in detecting the independent trigger type. Another interesting phenomenon is the instability of the MDR defense, where the ASR can sometimes be reduced to below 0.03% and at other times exceed 96%. Our analysis reveals that this is due to the MDR method possibly selecting the wrong community when identifying suspicious communities after partitioning, leading to a significant drop in defensive effectiveness. Therefore, these interesting situations are attributed to the trigger design or the defense methods, rather than our sample selection strategy. Nonetheless, regardless of the trigger type or defense method, these results verify that PMSB does not reduce resistance to defenses compared to random sample selection, and it still maintains the effectiveness of enhancing backdoor attacks.

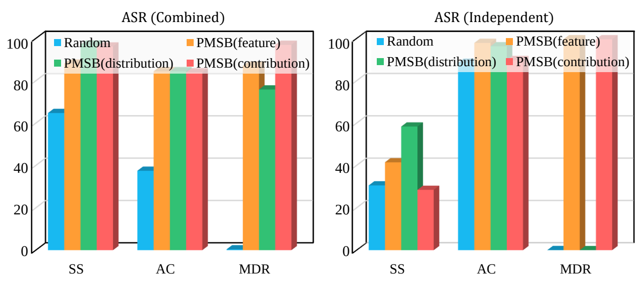

Figure 6: ASR after defense compared with vanilla attack.

6 Discussion

Possible Mitigations. While designing a novel adaptive defense is beyond our scope, we would like to discuss potential directions for eliminating poisoned samples. An intuitive possible defense strategy against PMSB could be to focus on the sample level, identifying and removing benign samples near the decision boundary. However, this poses a challenge for defenders in precisely identifying and removing poisoned samples. Removing too many samples may reduce model accuracy on clean tasks or cause overfitting, while removing too few may weaken backdoor defense. Another possible defense strategy is to analyze the differences in feature distributions. When triggers are embedded in samples, their feature distributions change, which enhances the contribution of trigger features to the model's predictions. Defenders could focus on analyzing the distribution differences of important decision-making features in the model to identify and remove poisoned samples. Further work is needed to validate these potential defense strategies.

Ethics and Responsible Code Release. In this paper, we did not attempt to test or poison any deployed malware detection systems for ethical considerations. Consistent with prior work on backdoor attacks against malware classifiers [35, 44], we responsibly release our code to support future research, particularly on defense methods. To prevent potential misuse, the code is hosted in a private repository, and requests will be verified before sharing.

7 Conclusion

In this work, we empirically analyze the reasons behind the poor performance of clean-label backdoor attacks under threat model constraints in the malware domain and present the results through visualizations. Although some methods have explored embedding triggers into hard-to-learn samples instead of random samples, they have achieved only limited gains in attack effectiveness. To address this issue, we propose a simple yet effective method: poisoning malware-similar benignware (PMSB) instead of random selection. This approach approximates the scenario of corrupted-label attack, thereby minimizing the interference of trigger characteristics by the original clean features, and enhancing the effectiveness of clean-label attacks. Additionally, we introduce three similarity measurement methods—feature-based distance, distribution-based distance, and contribution-based distance—to select malware-similar benignware. Extensive evaluations across three different trigger types and three datasets demonstrate the superiority and generality of PMSB.

References

[1] 2024. *AlienVault - Open Threat Exchange.* https://otx.alienvault.com/.

[2] 2024. *VirSCAN.org - Free Multi-Engine Online Virus Scanner.* https://www.virscan.org/.

[3] 2024. *VirusTotal.* https://www.virustotal.com/gui/home/upload/.

[4] Brandon Amos, Hamilton Turner, and Jules White. 2013. Applying machine learning classifiers to dynamic android malware detection at scale. In *2013 9th international wireless communications and mobile computing conference (IWCMC)*. IEEE, 1666–1671.

[5] Hyrum S Anderson and Phil Roth. 2018. Ember: an open dataset for training static pe malware machine learning models. *arXiv preprint arXiv:1804.04637* (2018).

[6] Mauro Barni, Kassem Kallas, and Benedetta Tondi. 2019. A new backdoor attack in cnns by training set corruption without label poisoning. In *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 101–105.

[7] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. 2013. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*. Springer, 160–172.

[8] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. 2018. Detecting backdoor attacks on deep neural networks by activation clustering. *arXiv preprint arXiv:1811.03728* (2018).

[9] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526* (2017).

[10] Edward Chou, Florian Tramer, and Giancarlo Pellegrino. 2020. Sentinet: Detecting localized universal attacks against deep learning systems. In *2020 IEEE Security and Privacy Workshops (SPW)*. IEEE, 48–54.

[11] Zheng Leong Chua, Shiqi Shen, Prateek Saxena, and Zhenkai Liang. 2017. Neural nets can learn function type signatures from binaries. In *26th USENIX Security Symposium (USENIX Security 17)*. 99–116.

[12] Corinna Cortes. 1995. Support-Vector Networks. *Machine Learning* (1995).

[13] Muhammad Al Fatih Abil Fida, Tohari Ahmad, and Maurice Ntahobari. 2021. Variance threshold as early screening to Boruta feature selection for intrusion detection system. In *2021 13th International Conference on Information & Communication Technology and System (ICTS)*. IEEE, 46–50.

[14] Yinghua Gao, Yiming Li, Linghui Zhu, Dongxian Wu, Yong Jiang, and Shu-Tao Xia. 2023. Not all samples are born equal: Towards effective clean-label backdoor attacks. *Pattern Recognition* 139 (2023), 109512.

[15] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C Ranasinghe, and Surya Nepal. 2019. Strip: A defence against trojan attacks on deep neural networks. In *Proceedings of the 35th annual computer security applications conference*. 113–125.

[16] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2017. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733* (2017).

[17] Richard Harang and Ethan M Rudd. 2020. SOREL-20M: A large scale benchmark dataset for malicious PE detection. *arXiv preprint arXiv:2012.07634* (2020).

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[19] Minqi Jiang, Songqiao Han, and Hailiang Huang. 2023. Anomaly detection with score distribution discrimination. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 984–996.

[20] Wenbo Jiang, Hongwei Li, Guowen Xu, and Tianwei Zhang. 2023. Color backdoor: A robust poisoning attack in color space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8133–8142.

[21] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems* 30 (2017).

[22] Dhilung Kirat and Giovanni Vigna. 2015. Malgene: Automatic extraction of malware analysis evasion signature. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 769–780.

[23] Shaofeng Li, Minhui Xue, Benjamin Zi Hao Zhao, Haojin Zhu, and Xinpeng Zhang. 2020. Invisible backdoor attacks on deep neural networks via steganography and regularization. *IEEE Transactions on Dependable and Secure Computing* 18, 5 (2020), 2088–2105.

[24] Yiming Li, Yang Bai, Yong Jiang, Yong Yang, Shu-Tao Xia, and Bo Li. 2022. Untargeted backdoor watermark: Towards harmless and stealthy dataset copyright protection. *Advances in Neural Information Processing Systems* 35 (2022), 13238–13250.

[25] Junyu Lin, Lei Xu, Yingqi Liu, and Xiangyu Zhang. 2020. Composite backdoor attack for deep neural network by mixing existing benign features. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 113–131.

[26] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2018. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International symposium on research in attacks, intrusions, and defenses*. Springer, 273–294.

[27] Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. 2020. Reflection backdoor: A natural backdoor attack on deep neural networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X 16*. Springer, 182–199.

[28] Zihao Liu, Tianhao Wang, Mengdi Huai, and Chenglin Miao. 2024. Backdoor attacks via machine unlearning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 14115–14123.

[29] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems* 30 (2017).

[30] Enrico Mariconti, Lucky Onwuzurike, Panagiotis Andriotis, Emiliano De Cristofaro, Gordon J. Ross, and Gianluca Stringhini. 2017. MaMaDroid: Detecting Android Malware by Building Markov Chains of Behavioral Models. In *24th Annual Network and Distributed System Security Symposium, NDSS 2017, San Diego, California, USA, February 26 - March 1, 2017*. The Internet Society. https://www.ndss-symposium.org/ndss2017/ndss-2017-programme/mamadroid-detecting-android-malware-building-markov-chains-behavioral-models/

[31] Rui Min, Zeyu Qin, Li Shen, and Minhao Cheng. 2024. Towards stable backdoor purification through feature shift tuning. *Advances in Neural Information Processing Systems* 36 (2024).

[32] Tuan Anh Nguyen and Anh Tuan Tran. 2021. WaNet - Imperceptible Warping-based Backdoor Attack. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. https://openreview.net/forum?id=eEn8KTtJOx

[33] Hemant Rathore, Swati Agarwal, Sanjay K Sahay, and Mohit Sewak. 2018. Malware detection using machine learning and deep learning. In *Big Data Analytics: 6th International Conference, BDA 2018, Warangal, India, December 18–21, 2018, Proceedings 6*. Springer, 402–411.

[34] Giorgio Severi, Tim Leek, and Brendan Dolan-Gavitt. 2018. Malrec: compact full-trace malware recording for retrospective deep analysis. In *Detection of Intrusions and Malware, and Vulnerability Assessment: 15th International Conference, DIMVA 2018, Saclay, France, June 28–29, 2018, Proceedings 15*. Springer, 3–23.

[35] Giorgio Severi, Jim Meyer, Scott Coull, and Alina Oprea. 2021. {Explanation-Guided} backdoor poisoning attacks against malware classifiers. In *30th USENIX security symposium (USENIX security 21)*. 1487–1504.

[36] Shweta Sharma, C Rama Krishna, and Rakesh Kumar. 2021. RansomDroid: Forensic analysis and detection of Android Ransomware using unsupervised machine learning technique. *Forensic Science International: Digital Investigation* 37 (2021), 301168.

[37] Bing Sun, Jun Sun, Wayne Koh, and Jie Shi. 2024. Neural Network Semantic Backdoor Detection and Mitigation: A Causality-Based Approach. In *Proceedings of the 33rd USENIX Security Symposium. USENIX Association, San Francisco, CA, USA*.

[38] Brandon Tran, Jerry Li, and Aleksander Madry. 2018. Spectral signatures in backdoor attacks. *Advances in neural information processing systems* 31 (2018).

[39] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. 2019. Label-consistent backdoor attacks. *arXiv preprint arXiv:1912.02771* (2019).

[40] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. 2019. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE symposium on security and privacy (SP)*. IEEE, 707–723.

[41] Xutong Wang, Chaoge Liu, Xiaohui Hu, Zhi Wang, Jie Yin, and Xiang Cui. 2022. Make data reliable: An explanation-powered cleaning on malware dataset against backdoor poisoning attacks. In *Proceedings of the 38th Annual Computer Security Applications Conference*. 267–278.

[42] Dongxian Wu and Yisen Wang. 2021. Adversarial neuron pruning purifies backdoored deep models. *Advances in Neural Information Processing Systems* 34 (2021), 16913–16925.

[43] Mingfu Xue, Can He, Jian Wang, and Weiqiang Liu. 2020. One-to-N & N-to-One: Two advanced backdoor attacks against deep learning models. *IEEE Transactions on Dependable and Secure Computing* 19, 3 (2020), 1562–1578.

[44] Limin Yang, Zhi Chen, Jacopo Cortellazzi, Feargus Pendlebury, Kevin Tu, Fabio Pierazzi, Lorenzo Cavallaro, and Gang Wang. 2023. Jigsaw puzzle: Selective backdoor attack to subvert malware classifiers. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 719–736.

[45] Limin Yang, Arridhana Ciptadi, Ihar Laziuk, Ali Ahmadzadeh, and Gang Wang. 2021. BODMAS: An open dataset for learning based temporal analysis of PE malware. In *2021 IEEE Security and Privacy Workshops (SPW)*. IEEE, 78–84.

[46] Shihao Zhao, Xingjun Ma, Xiang Zheng, James Bailey, Jingjing Chen, and Yu-Gang Jiang. 2020. Clean-label backdoor attacks on video recognition models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 14443–14452.

[47] Mingli Zhu, Shaokui Wei, Li Shen, Yanbo Fan, and Baoyuan Wu. 2023. Enhancing fine-tuning based backdoor defense with sharpness-aware minimization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 4466–4477.